PLD Agile

Durée du PLD : 8 séances de 4 heures

Equipe pédagogique : V. Barrellon, E. Egyed-Zsigmond, P.-E. Portier, C. Solnon

1 Description du système

L'application est inspirée d'un projet réel, piloté par le Grand Lyon entre 2012 et 2015, et visant à optimiser la mobilité durable en ville (voir www.optimodlyon.com). Nous nous focaliserons ici sur la partie concernant le frêt urbain et l'optimisation des tournées de livraisons en ville.

Au lancement de l'application, l'utilisateur charge un fichier XML décrivant un plan de ville. Ce fichier donne la liste des intersections (avec, pour chaque intersection, ses coordonnées) et la liste des tronçons de rues (avec pour chaque tronçon, le nom de la rue, l'intersection de départ, l'intersection d'arrivée, la longueur en décimètres et la vitesse moyenne de circulation en décimètres par seconde).

Quand un plan est chargé, le système le visualise et l'utilisateur peut alors charger un fichier XML décrivant une demande de livraisons. Ce fichier donne l'adresse de l'entrepôt et l'heure de départ de l'entrepôt. Il liste ensuite les livraisons à effectuer. Chaque livraison a une durée (en secondes) et une adresse. Une livraison peut éventuellement avoir une plage horaire (une heure de début et une heure de fin entre lesquelles la livraison doit avoir lieu). Les adresses (de l'entrepôt et des livraisons) correspondent à des intersections du plan.

Une fois qu'une demande de livraisons est chargée, le système visualise la position de chaque livraison sur le plan. L'employé peut alors demander au système de calculer une tournée. Cette tournée doit partir de l'entrepot, passer par chaque livraison, s'arrêter le temps correspondant à la durée de la livraison, puis revenir à l'entrepôt. Lorsque les livraisons ont des plages horaires, le livreur doit arriver après le début de la plage et repartir avant la fin de la plage. La durée d'une tournée est égale au temps nécessaire pour parcourir l'ensemble de ses tronçons, plus la durée de toutes les livraisons (plus les temps d'attente dans le cas où une livraison à une plage horaire et où le livreur arrive avant le début de la plage horaire). La durée de la tournée doit être minimale. S'il n'existe pas de tournée respectant toutes les contraintes liées aux plages horaires, alors le système le signale à l'utilisateur, et la tournée n'est pas calculée.

La tournée calculée par le système est visualisée sur le plan. Le système affiche également la liste des livraisons, dans l'ordre de la tournée, avec pour chaque livraison les heures d'arrivée et de départ prévues. L'utilisateur peut alors modifier interactivement la tournée (supprimer des livraisons, échanger l'ordre de deux livraisons, ajouter de nouvelles livraisons, ou changer des plages horaires), et demander au système de modifier les horaires de passage en conséquence. Si une modification amène à ne plus respecter une plage horaire, alors le système le signale à l'utilisateur et la modification n'est pas effectuée. À tout moment, l'utilisateur peut demander l'annulation de modifications apportées à la tournée.

Lorsqu'une tournée a été validée, l'utilisateur peut demander la génération d'une feuille de route : une feuille de route est un fichier au format texte donnant la liste des livraisons à faire (dans l'ordre de passage de la tournée) et, pour chacune de ces livraisons, l'adresse de livraison, les heures prévues d'arrivée et de départ, et l'itinéraire à suivre pour rejoindre cette livraison depuis la livraison précédente ou depuis le dépôt.

2 Organisation

Le projet sera réalisé en hexanôme selon un processus de développement itératif. Le projet comportera au moins deux itérations, et la première itération durera 4 séances. Vous ferez une première démonstration de votre application à la fin de ces 4 séances. Lors des 4 séances suivantes, vous pourrez choisir de faire entre une et quatre itérations.

Première itération : Cette première itération correspond à la phase d'inception de la méthode UP. L'objectif est d'identifier les principaux cas d'utilisation, de concevoir une première architecture de votre

application, et d'implémenter quelques cas d'utilisation afin d'avoir un premier noyau exécutable sur lequel le client pourra vous faire un retour.

Les livrables qui vous seront demandés à l'issue des quatre premières séances sont :

- Modèle du domaine
- Glossaire
- Diagramme de cas d'utilisation
- Description textuelle structurée des cas d'utilisation qui ont été analysés et/ou implémentés
- Liste des événements utilisateur et diagramme Etats-transitions correspondant aux cas d'utilisation qui ont été analysés
- Diagramme de packages et de classes
- Diagramme de séquence du calcul de la tournée à partir d'une demande de livraisons
- Planning effectif de la première itération, détaillant pour chaque membre de l'équipe le temps passé sur chaque activité

Vous ferez une démonstration de votre prototype au début de la cinquième séance de projet.

Itérations suivantes : Au début de chaque itération suivante, vous choisirez les cas d'utilisation (ou les scénarios de cas d'utilisation) qui seront analysés, conçus et/ou implémentés pendant l'itération, en estimant pour chacun la durée prévue. A la fin de chaque itération, vous ferez une revue rapide pour récapituler ce qui a été réalisé en fonction des objectifs initiaux, et comparer les durées effectives avec les durées prévues.

Les livrables qui vous seront demandés à la fin du projet sont :

- Description textuelle structurée des cas d'utilisation
- Document expliquant les choix architecturaux et design patterns utilisés
- Code de votre application et des tests unitaires
- Documentation JavaDoc du code
- Diagrammes de packages et de classes retro-générés à partir du code
- Rapport sur la couverture des tests
- Pour chaque itération : planning prévisionnel et planning effectif

Environnement de développement : Nous vous proposons de développer en Java. Si vous souhaitez utiliser un autre langage orienté objet, vous devrez demander notre accord au préalable. Vous utiliserez un environnement de développement intégré (IDE) et des outils pour automatiser les tests unitaires, évaluer la couverture des tests, favoriser le travail collaboratif et la gestion des versions, et générer la documentation en ligne du code. Vous pourrez notamment utiliser l'IDE Eclipse [1] et ses plug-in ObjectAid [2], pour re-générer un diagramme de classes à partir du code Java, JUnit [3], pour automatiser la réalisation des tests unitaires, et EclEmma [4], pour évaluer la couverture des tests.

Vous pouvez utiliser la version démo de StarUML [5] pour saisir des diagrammes UML (diagrammes de classes, de packages, de séquences et états-transitions).

Vous générerez une documentation en ligne de votre code en utilisant JavaDoc, et vous suivrez le code de style préconisé par Oracle [6].

Pour calculer la tournée, vous pouvez utiliser le code Java disponible sur Moodle. La classe `TSP1` implémente la variante la plus basique des algorithmes vus en cours. Vous pouvez programmer des variantes plus élaborées en redéfinissant les méthodes `bound` et `iterator`.

1. `http://help.eclipse.org/juno/index.jsp` (voir le Java Development User Guide)
2. `http://www.objectaid.com/`
3. `http://www.junit.org/`
4. `http://www.eclemma.org/`
5. `http://staruml.io/`
6. `http://www.oracle.com/technetwork/java/codeconventions-150003.pdf`